REDUCTION OF AUTODIN/NIDN LINE DISCIPLINES TO PROGRAMMED CONTROL LOGIC

Jane Frances Renninger

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

REDUCTION OF AUTODIN/NIDN LINE DISCIPLINES

TO PROGRAMMED CONTROL LOGIC

by

Jane F. Renninger

June 1975

Thesis Advisor: V. M. Powers

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Reduction of AUTODIN/NIDN Line Disciplines to Programmed Control Logic	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis; June 1975
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Jane F. Renninger	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93940	12. REPORT DATE: June 1975
	13. NUMBER OF PAGES: 69
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Naval Postgraduate School, Monterey, California 93940	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

AUTODIN
NIDN
communication control logic

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this design of a backup control for the OSIS Communication Processor, the AUTODIN/NIDN line disciplines were analyzed and reduced to programmed control logic. The method of approach was developed then applied to the AUTODIN/NIDN protocol. A definition of "state," as used in the context of the thesis, is presented, followed by detailed state transition tables and state diagrams. An instruction set of only three instructions

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. (cont.)

was designed which can accomplish all the necessary tasks of the control logic. This instruction set was then used to program the Receive and Transmit control functions of the OSIS system. An implementation plan is presented to suggest possible ways the control unit can be made operational.

A Reduction of AUTODIN/NIDN Line Disciplines to Programmed Control Logic

by

Jane Frances Renninger
Lieutenant Commander, United States Navy
B.S., Pennsylvania State University, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1975

ABSTRACT

In this design of a backup control for the OSIS Communication Processor, the AUTODIN/NIDN line disciplines were analyzed and reduced to programmed control logic. The method of approach was developed then applied to the AUTODIN/NIDN protocol. A definition of "state", as used in the context of the thesis, is presented, followed by detailed state transition tables and state diagrams. An instruction set of only three instructions was designed which can accomplish all the necessary tasks of the control logic. This instruction set was then used to program the Receive and Transmit control functions of the OSIS system. An implementation plan is presented to suggest possible ways the control unit can be made operational.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

TABLE OF ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
ASC	Automatic Switching Center
AUTODIN	AUTOmatic Digital Network
DCS	Defense Communications System
DSTE	Digital Subscriber Terminal Equipment
NIDN	Navy Intelligence Data Network
OCP	OSIS Communication Processor
OSIS	Ocean Surveillance Information System

CONTROL CHARACTERS:

ACK	ACKnowledge
BP	Block Parity
CAN	CANcel
DEL	DELete
EM	End of Medium
ETB	End of Text Block
ETX	End of Text
INV	Suspected INValid Message
MC	Mode Change
NAK	Negative ACKnowledge
REP	REPly
RM	Reject your Message
SEL	SELection channel
SOH	Start Of Header
STX	Start of Text
SYN	SYNchronous Idle
WBT	Wait Before Transmitting

I. INTRODUCTION

A. PURPOSE

This thesis is a design study to identify a method of approach and outline an implementation scheme for developing a backup control for the Ocean Surveillance Information System (OSIS) Communication Processor (OCP). The OCP is currently under development by the Naval Electronics Laboratory Center (NELC), San Diego. The main task in developing a backup for OCP is the reduction of the OSIS communication line disciplines to programmed control logic.

B. SCOPE

There appear to be many names for communication line disciplines which have only slight variation among them. The method of approach developed for reducing the OSIS protocol could also be used to reduce other similar communication disciplines to programmed control logic.

The OSIS data is transmitted and received via the Navy Intelligence Data Network (NIDN) circuits. These circuits operate with a modified format of the Defense Communications System (DCS) Automatic Digital Network (AUTODIN). Therefore, this document reflects the AUTODIN terminology and specifications with exceptions noted as they apply to NIDN.

II. GENERAL DESCRIPTION OF AUTODIN

The DCS AUTODIN is a world-wide digital communications network of Automatic Switching Centers (ASC's) and computer interface subscriber terminals in which both synchronous and asynchronous operation is employed in the exchange of messages. Since the OCP initial operating capability operates in accordance with AUTODIN Mode I continuous procedure as described in reference [2] and modified by reference [10], the AUTODIN specifications discussed will be limited to those pertaining to Mode I continuous operation. Mode I prescribes duplex synchronous operation with automatic error and channel controls allowing independent and simultaneous two-way operation.

The AUTODIN line discipline for synchronous operation requires that messages be transmitted in blocks of American Standard Code for Information Interchange (ASCII) coded characters. "All characters will use seven bits for information and an eighth bit for parity. Characters are transmitted serially by bit with the low order bit first and the parity bit last. Message characters will have odd parity; control characters will have even parity with the exception of the last character of each block called the Block Parity (BP) character which may have odd or even parity. Characters having even parity, which are not recognized as one of the assigned control codes, will be treated as errors." [2]

The message formats are as described in JANAP 128 (E) [4] The typical message contains three parts: the header block or blocks, a variable number of text blocks, and a single end-of-message block. Each block must be preceded by

and followed by two framing control characters, the last of which is the BP character. The text may contain as many as 80 odd parity message characters. If 79 or fewer message characters are transmitted or received, the text must end with the special control character End-of-Medium (EM). Other even parity control characters may be embedded in a message block, including transmit and receive character pairs, the "invalid" sequence (INV), and the "mode change" (MC) character. Of these, only the MC character adds to the text character count.

Figures 1 and 2 show the formats for messages containing 80 text characters and 79 or fewer text characters, respectively.

First Framing Control Char	Second Framing Control Char	80 Text Chars	Third Framing Control char	Fourth Framing Control char

FIGURE 1.

First Framing Control Char	Second Framing Control Char	79 or Fewer Text Chars	EM	Third Framing Control Char	Fourth Framing Control Char

FIGURE 2.

There are five types of control characters:

1. Communication Framing Characters. The positional and coding significance of these characters serve to delineate the beginning and end of each serial block of data in a message. The framing characters are: Start of Heading (SOH), Select (SEL), Start of Text (STX), Delete (DEL), AUTODIN Security (SEC), End of Transmission Block (ETB), End of Text (ETX), and Block Parity (BP).

2. Receive Control Characters. Receive control characters are answers or responses to blocks or control sequences which have been received. They are transmitted in identical contiguous pairs, and may be interspersed anywhere in the bit stream except between two adjacent framing characters. Their parity is not added to the block parity sum nor are they added to the text count. The receive control characters are Acknowledge Number One (ACK1), Acknowledge Number Two (ACK2), Negative Acknowledge (NAK), Reject Your Message (RM), and Wait Before Transmitting (WBT).

3. Transmit Control Characters. These characters are sent by the transmitting station to direct the receiving station to take some action. They are transmitted in identical contiguous pairs and, for continuous operation, may be sent between blocks or between the text and third framing character. The transmit control characters are Reply (REP) and Cancel (CAN).

4. Special Control Characters. There are three special control characters, each of which performs a unique function. They are as follows:
   a. End of Medium (EM). EM marks the end of text in a block containing 79 or fewer

characters. EM is included in the block parity sum.

b. Mode Change (MC). MC marks the end of the binary text portion of a message.

c. Suspected Invalid Message (INV). INV is sent when an unsolicited answer is received. It is transmitted as a two-character sequence and may be interspersed anywhere in the bit stream except between two adjacent framing characters or between adjacent characters of a two-character control sequence. An alarm is activated whenever the INV sequence is transmitted or received.

5. Synchronous Idle (SYN). SYN is used with synchronous operation to enable character synchronization of the bit stream between the transmitter and receiver. SYN is transmitted continuously whenever the transmitter cannot or should not transmit data.

For a detailed description of each control character see reference [2].

III. REDUCTION ALGORITHM

The method of approach developed and used for reducing the OSIS communication line disciplines is as follows:

1. Define the parameters and limits of the system.
2. Analyze the line discipline specifications, especially noting method of control, sequence of events, causes for changing state, error procedures, and exceptions to any rule.
3. Construct very general state diagrams for both the transmit and receive functions in order to identify the general flow of the system.
4. Write descriptions of the functions performed during each state and the conditions under which a change of state is made.
5. Construct state transition tables.
6. Construct detailed state diagrams including dummy states, where necessary, to show sub-condition decision points.
7. Determine necessary flags, indicators, counters, and timers.
8. Define or select the instruction set necessary for implementation.
9. Code the protocol using the defined instruction set.

IV. APPLICATION OF ALGORITHM

A. PARAMETERS AND LIMITS

As noted earlier, only the AUTODIN Mode I continuous operation will be described in this thesis. Other parameters and assumptions of the system also limit the scope of the analysis and design. First of all, the OCP backup will be acting as a subscriber terminal rather than an ASC. The single block message, non-standard code message, and variable length record will not be addressed or implemented. The initial state of the system will be assumed to be "in character frame." [2] Since the OCP backup will be a Digital Subscriber Terminal Equipment (DSTE) type of backup, it will not be required to send the Reject Your Message (RM) response. Thus, RM has been eliminated as a consideration for transmission but will be accounted for as a possible received character. The Mode Change (MC) and Suspected Invalid Message (INV) Special Control characters are not used by OSIS.

B. ANALYSIS OF AUTODIN/NIDN

A terminal has essentially two functions --- transmitting and receiving. Each function works hand-in-hand with the other to produce effective communications. The transmitter properly formats blocks of data, sends control characters designated by the receiver, and interprets and responds to Receive Control characters. On the other hand, the receiver upon request sends replies via its transmitter, informs the transmitter upon receipt of a Receive Control character, removes control and framing characters, and stores text. The Transmit and

Receive functions communicate with each other through the use of flag flip-flops, counters, and answer timers. Error detection is on the basis of character parity and block parity. Error correction is accomplished by retransmission of blocks in which errors were detected.

1. Analysis of Receive. In continuous operation, character synchronization must be established before message reception can begin. Once synchronization has been established, loss of synchronization will occur when the receiver times out and fails to detect the SYN character. When this happens, character synchronization must be re-established before the receiver is ready to continue accepting messages. Message reception once begun is continuous, with the SOH/STX character of a block under normal conditions being contiguous to the BP character of the previous block. Appropriate character patterns are expected at certain positions within the message.

In the start of block or first framing control character position, the receiver accepts SYN characters, Receive Control characters, Transmit Control characters, or the proper SOH or STX character. The SOH character is the first framing character of the first block of a message. The receiver expects the STX character in the first framing position of each succeeding block until either the ETX character or CAN sequence has been received and acknowledged, whereupon the SOH character is again expected. If the previous block is to be acknowledged with NAK, the receiver will ignore a valid SOH or STX character, if received immediately after the BP character of the block in error, and return to the start of block position.

The acceptable characters in the second framing control character position are the even parity SEL characters "H" or "D" for the first block of a message and the DEL character

for all succeeding message blocks. If the second character is not even parity, the receiver will ignore the block and wait for the REP or CAN sequence. While AUTODIN uses any one of thirteen SEL characters to denote the output channel compatible with the message format (cards or tape), OSIS uses only the SEL characters "H" and "D". The H character signifies Record Traffic and the D character denotes Bulk Data and OPSCOMM (analyst-to-analyst information exchange) messages. If the SEL character received is not H or D the "SEND NAK" flag is set.

In the text portion of the message, only odd parity data characters, even parity Receive Control characters, and the even parity Special Control character EM are allowed. The Receive Control characters must be double character sequences. The SEND NAK flag is set if a double character sequence is broken or if an even parity data character or even parity Transmit Control character is received. When the SEND NAK flag is set, it will not be used to transmit a NAK response until after receipt of the BP character for that block. If a framing control character or a SYN character is received in the text position, the receiver will ignore the block, go to the waiting state, and wait for the REP or CAN sequence.

Reception of text will continue until either a total of 80 data characters or the even parity EM character is received. The receiver will then proceed to the third framing control character or end of block position, in which the acceptable characters are ETB, ETX, SYN, Receive Control characters, or Transmit Control characters. (The situation of receiving SYN or Transmit Control character sequences, before receiving the end of block framing character, is encountered when the transmitting station has not received an answer for the previous block.) After an ETB or ETX character is received and accepted, the next character,

regardless of what it actually is, will be accepted as the BP character and will be compared to the block parity calculated by the receiver. If they agree and the SEND NAK flag has not been previously set, the block is accepted, the proper "SEND ACK (1 or 2)" flag is set for the transmitter, and the receiver examines the start of block framing position of the next block. If the local SEND NAK flag has been set, the receiver sets the NAK flag for the transmitter, then returns for the start of block framing position of the same block which will be retransmitted.

Whenever a block is accepted, it must be answered. If the receiver temporarily cannot accept any additional blocks, it may halt the distant receiver by answering the received block with the WBT sequence. If this delay in ability to accept another block continues, then the next sequence expected is the REP sequence. Upon receiving the REP sequence, the receiver will answer with the WBT sequence, and this will continue until the receiver is able to accept another block. When the receiver is ready to accept blocks again, it will answer the REP sequence with the acknowledgment for the last block it accepted.

2. Analysis of Transmit. The transmitter checks the flags to determine if any Receive Control characters or any Transmit Control characters are to be sent in the start of block position. If so, the transmitter will send these double character sequences contiguously. If there is a block to send, SOH or STX will be sent for the first or succeeding blocks, respectively; otherwise, SYN characters will be sent.

If SOH is sent, the transmitter will send a SEL character of "H" or "D" for the second framing control character, depending on whether the message is Record Traffic or Bulk Data/OPSCOMM. If STX is sent, the

transmitter follows it with DEL for the second framing control character.

In the text portion of the message, Receive Control characters, text characters, or the Special Control character EM may be sent. Only text characters and the EM character are used to increment the text counter. After transmitting 80 text characters, or fewer than 80 text characters plus the EM character, the transmitter will check for an answer to the previous block. If an answer has not been received, the transmitter will send SYN characters until the answer timer expires. (Normally the answer timer would expire before the last data character of the following block is sent.) If WBT is received or the answer timer expires before the 80th text character or EM character is transmitted, the transmitter will wait until the 80th character or EM is sent, then send the REP sequence. If an answer to the previous block is received and it is the proper ACK sequence, the transmitter examines the end of block or third framing control character position. If it is not the proper ACK sequence the previous block will be retransmitted.

In the end of block position, if any answer flags have been set by the receiver, the transmitter will send the answer. If there is no answer to be sent, the ETB character (or ETX character in the case of last message block) will be sent, followed by the calculated block parity character. The transmitter then starts sending the next block, if there is one; otherwise, it sends SYN characters.

When waiting for the answer to the previous block or the answer to a transmitted CAN sequence, the transmitter sends any Receive Control or Transmit Control characters that have been flagged to be sent. If there are no characters to be sent and the answer timer expires, the transmitter will

send the proper Transmit Control character. If the CAN sequence is acknowledged, the transmitter will proceed to process the start of block position. If the last block is properly acknowledged, the transmitter will process the end of block position.

In synchronous operation the ACK's are alternated, first ACK1 then ACK2 for alternate blocks. The remote receiver responds with, and the local transmitter expects, alternate ACK's as answers for transmitted blocks. The CAN sequence must always be answered with the ACK2 sequence. After the last block of a message has been acknowledged, the transmitter may send the CAN sequence, forcing the remote receiver to acknowledge with ACK2. This merely serves as an ACK reset, requiring the remote receiver to respond to the first block of the next message with ACK1.

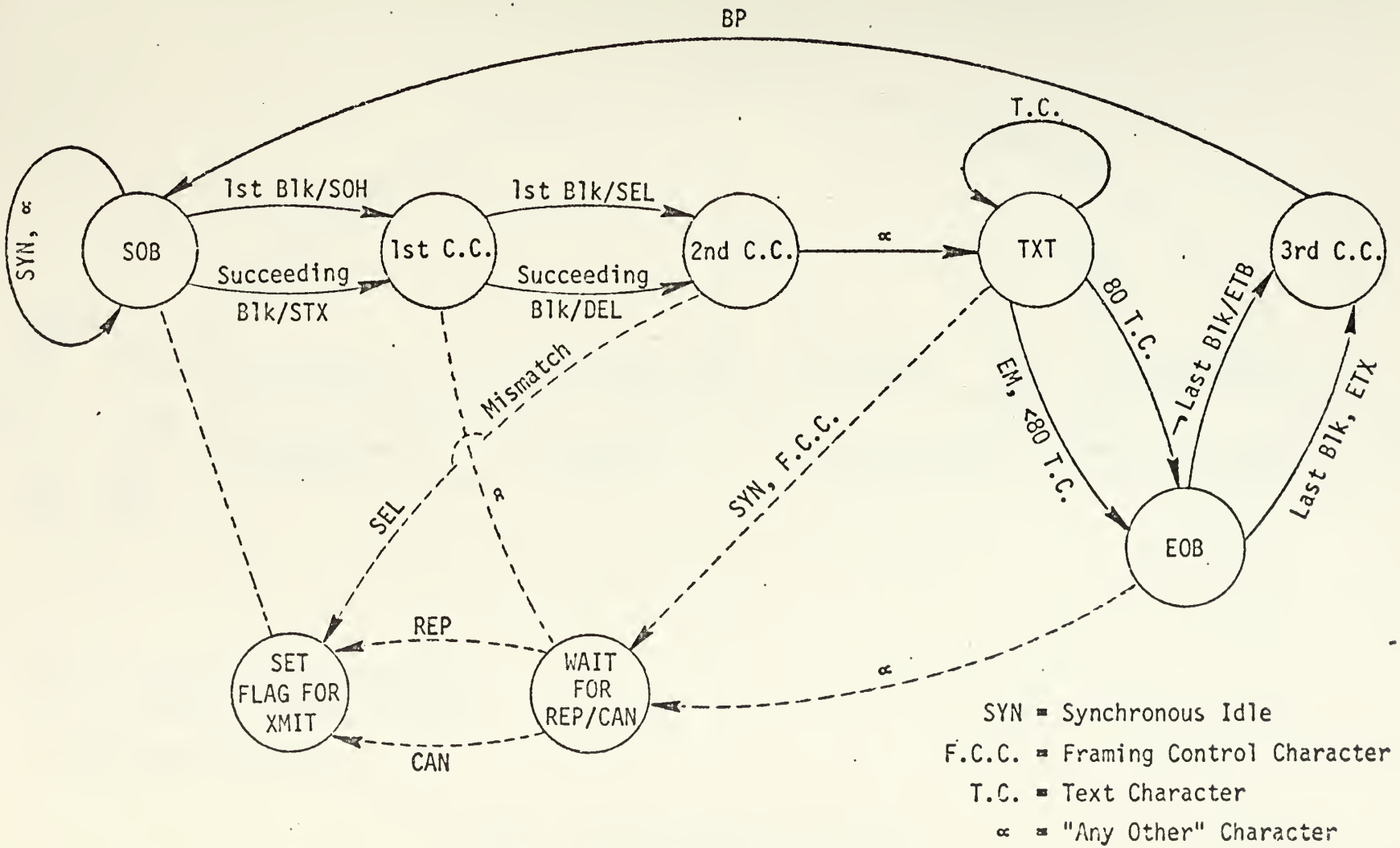
The CAN and REP sequence timing procedures are identical. A CAN sequence is sent if the remote receiver rejects a message or if there is an uncorrectable error condition at the transmitter. A REP sequence is sent when an answer for a transmitted block is not received in the allowable response time. Each CAN or REP sequence will be repeated by the terminal up to three times if no reply is received. If no answer is received after the CAN/REP sequence has been sent three times, a "no reply" alarm is activated and the terminal will continue to send the CAN/REP sequence at proper intervals. When the CAN sequence is properly answered with ACK2, the next block is transmitted. If the previous ACK sequence is received in response to a REP sequence, the present block will be retransmitted.

3.General Flow of the System. The transition through the Receive and Transmit functions is very generally presented by the state diagrams in Figures 3 and 4,

respectively. They are "general" in the sense that specific flags, conditions, tasks, and subconditions are not included. Only the normal "main line" flow is shown.

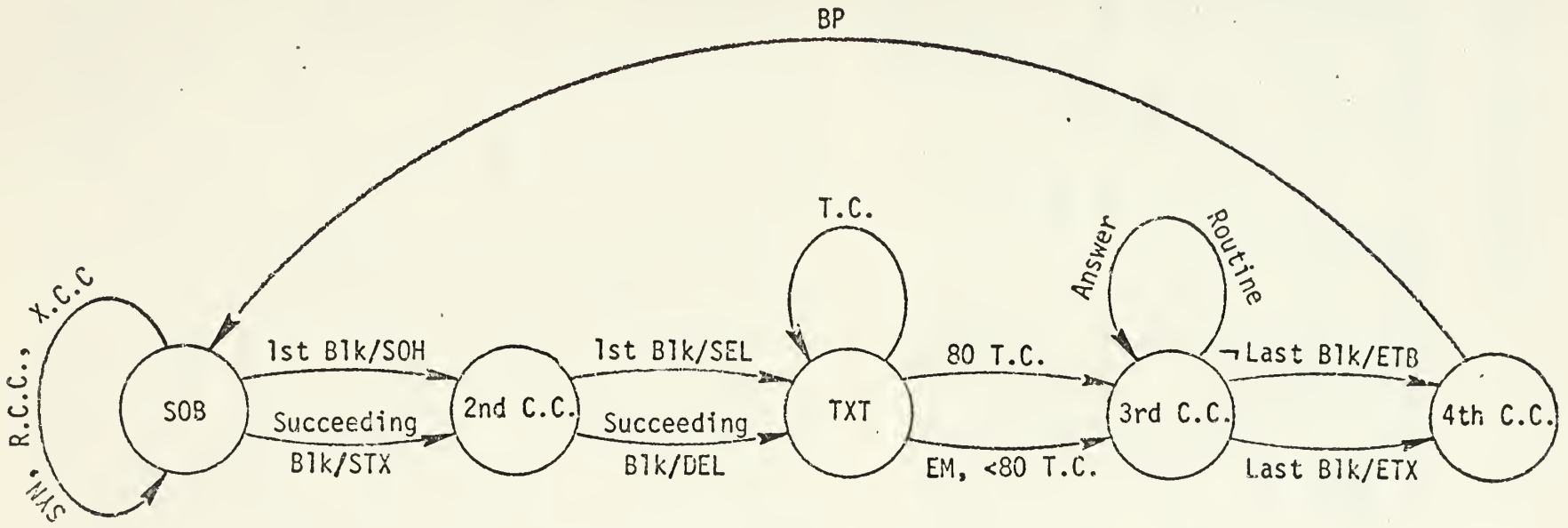
In the Receive diagram the reception of a correct and expected control or text character causes transition to the state which handles that character. An incorrect character forces the Receive function to go to a wait state, then to set flags for the transmitter and return to the start of block (SOB) state.

The Transmit diagram is slightly different in that the name of the state indicates the type of character it sends.



MODE I, RECEIVE (General Diagram)

FIGURE 3.



R.C.C. = Receive Control Character
X.C.C. = Transmit Control Character
T.C. = Text Character

MODE I, TRANSMIT (General Diagram)

FIGURE 4.

C. STATE DESCRIPTIONS

According to Webster's New World Dictionary a state is defined to be "a set of circumstances or attributes characterizing a (particular) thing at a given time; a condition." In the context of this thesis, a state is a condition which exists at a point in time during message transmission or reception. This condition or state depends on the previous message state, the message characters received as input, and the existing condition of memory (e.g. flags, timers, and alarms). This state and existing conditions uniquely determine the output message character (for transmit), the tasks to be performed before proceeding to the next state, and the next Transmit or Receive state to be entered.

A colloquial use of the word "state" appears in descriptions of communication protocols. The AUTODIN/NIDN systems have six Major States to be considered:First Framing Character State (S1), Second Framing Character State (S2), Text State (S3), Third Framing Character State (S4), Block Parity (S5), and Special Control State (S6). (Hereafter, these states will be referred to by their number.) The name of each of these Major States indicates the type of message character that is to be transmitted or received by that particular state. The particular message character to be transmitted or received by the Major State is determined by the block of the message that is being processed --- the first block, an intermediate block, the last block, or no block (in the case of a WAIT/CONTROL situation). In effect, the blocks represent Message States. The normal processing of each block transitions through the Major States S1 to S5. The WAIT/CONTROL or no-block Message State coincides with the Major State S6. Therefore, calculating the various

combinations of Message States and Major States for both the Receive and Transmit functions, there are actually thirty-two states in the AUTODIN/NIDN systems. Figure 5 is a matrix showing the relationship between the Major States and Message States, and giving the specific character to be transmitted or received. (The "x" represents other control characters or text.)

	MAJOR STATE					
	S1	S2	S3	S4	S5	S6
MESSAGE STATE						
FIRST BLOCK	SOH	SEL	X	ETB	BP	
INTERMEDIATE BLOCK	STX	DEL	X	ETB	BP	
LAST BLOCK	STX	DEL	X	ETX	BP	
WAIT/CONTROL						x

FIGURE 5.

Since the Major State transitions are essentially the same for each block, this document will emphasis the Major States, with exceptions noted as they pertain to a particular Message State.

The transition description lists in the next section of this thesis describe in detail the states, possible conditions that could exist in each state, the tasks prescribed for each condition, and the next Major State to be entered. In order to complete the specification of conditions and actions for transitions among such states, "dummy" states have been introduced. A dummy state is merely a sub-condition decision point. The dummy states ease the implementation and diagram presentation of the state transitions. Therefore, the set of total states, a description of the history of the system which is necessarily and sufficiently precise to establish the proper

action for each condition, is comprised of the Message States and Major States together with the dummy states.

D. TRANSITION DESCRIPTIONS

Using the method of approach defined earlier, Tables I and II were constructed to describe the tasks performed during each state and the conditions causing a change of state for the Receive and Transmit functions, respectively. Figures 6 and 7 are the resulting state diagrams which present a visual picture of the transition flow. (These tables and figures begin on page 29.) Generally speaking, the Receive table and diagram is more complex, since every possible situation must be considered when receiving data to be sure all errors are detected. The Transmit function, on the other hand, is in control of the situation and consequently the tasks to be performed during each state are simple and straightforward. Only the Transmit function's WAIT/CONTROL state (S6) has complex tasks. This is because it must be able to act on any possible response or lack of response from the remote terminal or ASC.

The reader will notice that the transition flow of the Transmit function is somewhat different from the transition flow through the Receive function. As long as the receiver continues to receive the characters it expects, with their proper parity, it will move through Major States S1 to S5, and back to S1 for the start of the following block. If the receiver detects a parity error or a Framing Control character it doesn't expect, it will set a flag to wait for a REP or CAN sequence and will go to the WAIT state, S6. It will also set the WAIT flag and go to S6 if the last received block was not acknowledged prior to receiving the BP character of the present block or if the

buffer is full when leaving state S5. The transmitter moves through Major States S1 to S3, then goes off to the WAIT/CONTROL state, S6, to check for an answer to the previous block. If it has not been received, it waits for it. If a proper acknowledge is received, the transmitter returns to the "main line" flow by going to S4, on to S5, and then back to S1 for the next block. If a proper acknowledgment is not received, the transmitter will return to S1 and retransmit the previous block. In the Transmit function the WAIT/CONTROL state can also be entered from S1, if an RM character has been received and requires the transmission of a CAN sequence.

The reasons for leaving the WAIT/CONTROL state are also different between the Receive and Transmit functions. In the Receive function, the only condition which allows transition out of Receive state S6 is the clearing of the WAIT flag. And then, the transition is always to Receive state S1. While in the Transmit function, there are several conditions for leaving Transmit state S6. An incorrect acknowledge, a negative acknowledge, and a response to a cancel sequence will all cause a transition from Transmit state S6 to Transmit state S1. However, a correct and expected acknowledge will cause transition to Transmit state S4.

As mentioned in the previous section, the block of the message being processed determines the particular control characters to be transmitted or received. These Message States are regulated by flags. In the Receive function if the "CAN/ETX RECEIVED LAST" flag is set and in the Transmit function if the "ETX SENT LAST" flag is set, it means that the First Block is being processed. If these flags are clear, an Intermediate or Last Block is being processed. This distinction is sufficient for sending or receiving the first two control characters by Major States

S1 and S2.

The reasons for entering and leaving the WAIT/CONTROL Message State were presented above. The Major States S3 and S5 are the same for the First, Intermediate, and Last Blocks. That leaves only the relationship between the Major State S4 and the first three Message States to be accounted for. In this regard, it is sufficient for the Transmit or Receive function to know only if it is processing a Last Block or not processing a Last Block, since the functions treat the First and Intermediate Blocks the same in Major State S4. A determination that the Last Block is being processed is made by checking a flag set by the host computer.

Two other important "toggle switch" flags help maintain message integrity. They are the "SEND ACK1" and the "ACK1 EXP'D" flags. The first is used by the Receive function to tell the Transmit function which acknowledgement to send in response to a received block. The second is used by the Transmit function to keep track of which response it expects from the remote receiver in regards to a particular transmitted block or a CAN sequence. When these flags are not set, it implicitly means "send ACK2" and "ACK2 is expected".

TABLE I.

TRANSITION DESCRIPTION FOR RECEIVE

STATE	CONDITION	TASKS	NEXT STATE
S1	----	Receive a character	
	Odd parity	Ignore character	S1
	Character = SYN	Increment SYNCNT	S11
	Character ≠ SYN	Clear SYNCNT	
	Character = REP/CAN/ACK1/ACK2 NAK/WBT/RM	Store character	
		Receive a character	S12
	Character = SOH	Inhibit Answer Timer	S13
	Character = STX	Inhibit Answer Timer	S14
	Any other even char	Ignore character	S1
S11	SYNCNT ≠ 4	----	S1
	SYNCNT = 4	Start Answer Timer	S1
S12	Character ≠ Previous Character	Set "SEND NAK" flag	S1
	Character = REP	Perform REP routine	S1
	Character = CAN	Perform CAN routine	S1
	Character = ACK1/ACK2/NAK/WBT/RM	Perform RECEIVE CONTROL routine	S1

RECEIVE DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S13	"NAKFLG" set	Ignore character	S1
	"CAN/ETX REC'D LAST" flag set	Accept character	S2
	"CAN/ETX REC'D LAST" not set	Ignore character	S1
S14	"NAKFLG" set	Ignore character	S1
	"CAN/ETX REC'D LAST" flag not set	Accept character	S2
	"CAN/ETX REC'D LAST" flag set	Ignore character	S1
S2	----	Receive a character	
	Odd parity	Ignore character	
		Set "WAIT FOR REP/CAN" flag	
		Start Answer Timer	S6
	"CAN/ETX REC'D LAST" flag set	Add character to BP	
		Clear TEXTCOUNT	S21
	Character = DEL	Add character to BP	
		Clear TEXTCOUNT	
		Accept character	S3
	Any other even character	Set "SEND NAK" flag	S3
S21	Character = "H" or "D"	Accept character	S3
	Character ≠ "H" or "D"	Set "SEND NAK" flag	S3

RECEIVE DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S3	----	Receive a character	
	Odd parity	Increment TEXTCOUNT	
		Add character to BP	
		Accept character	S31
	Character = ACK1/ACK2/NAK/WBT/RM	Store character	
		Receive a character	S32
	Character = EM	Add character to BP	
		Accept character	S4
	Character = SOH/STX/DEL/H/D/ ETX/ETB/BP	Set "WAIT FOR REP/CAN" flag	
		Start Answer Timer	S6
	Any other even character	Set "SEND NAK" flag	S3
S31	TEXTCOUNT < 80	----	S3
	TEXTCOUNT = 80	----	S4
S32	Character ≠ Previous Character	Set "SEND NAK" flag	S3
	Character = ACK1/ACK2/NAK/WBT/RM	Perform RECEIVE CONTROL routine	S3

RECEIVE DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S4	----	Receive a character	
	Odd parity	Set "WAIT FOR REP/CAN" flag	
		Start Answer Timer	S6
	Character = SYN	Ignore character	S4
	Character = REP/CAN/ACK1/ACK2 NAK/WBT/RM	Store character	
		Receive a character	S41
	Character = SOH/SEL/STX/DEL	Set "WAIT FOR REP/CAN" flag	
		Start Answer Timer	S6
	Character =ETB	Add ETB character to BP	
		Clear "CAN/ETX REC'D LAST" flag	S42
	Character =ETX	Add ETX character to BP	
		Set "CAN/ETX REC'D LAST" flag	S42
S41	Character ≠ Previous Character	Set "SEND NAK" flag	S4
	Character =REP	Perform REP routine	S4
	Character = CAN	Perform CAN routine	S4
	Character = ACK1/ACK2/NAK/WBT/RM	Perform RECEIVE CONTROL routine	S4

RECEIVE DESCRIPTION (CONTINUED)

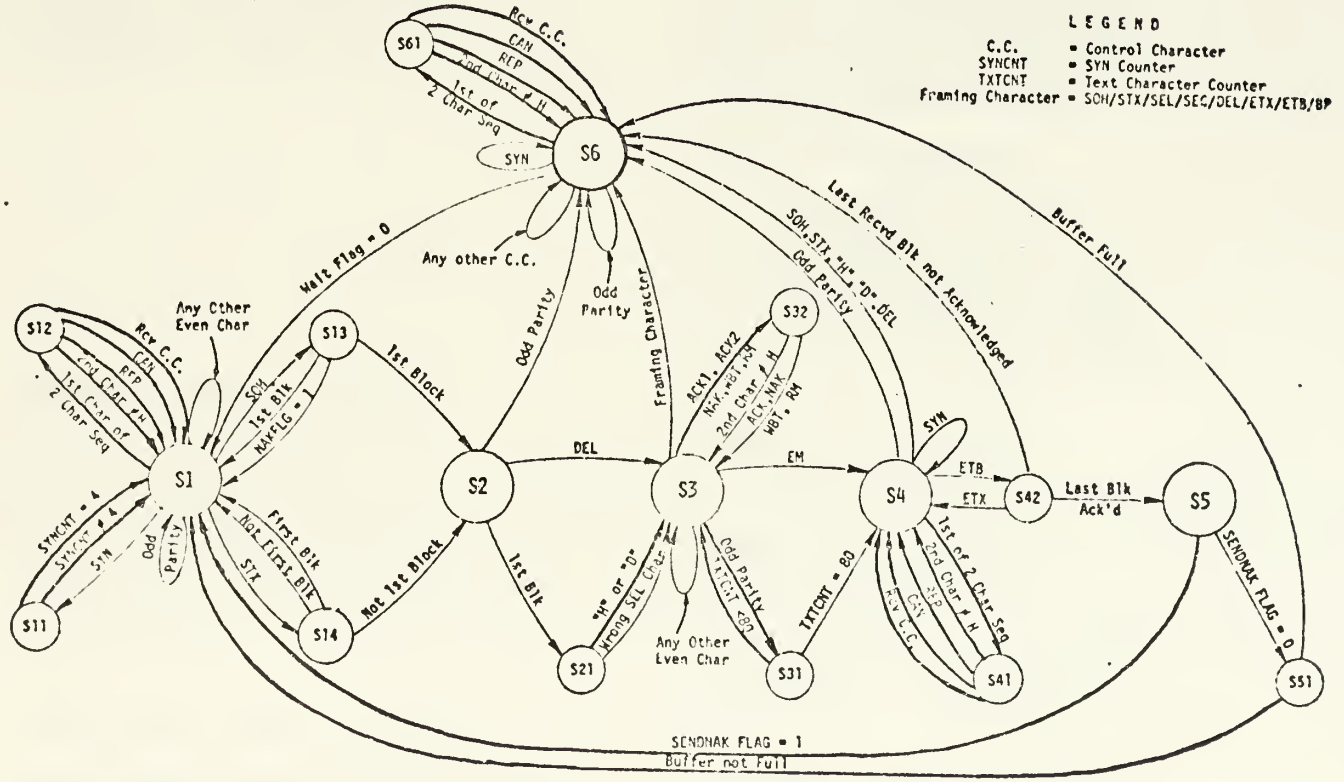
STATE	CONDITION	TASKS	NEXT STATE
S42	"LAST RECD BLK ACK'D" flag set	Accept character	
		Clear "LAST RECD BLK ACK'D" flag	S5
	"LAST RECD BLK ACK'D" not set	Set "WAIT FOR REP/CAN" flag	
		Start Answer Timer	S6
S5	----	Receive a character	
	Received BP ≠ Calculated BP	Set "SEND NAK" flag	
	----	Clear Calculated BP	
	"SEND NAK" flag set	Dump buffer	
		Set "NAKFLG" for Transmitter	
		Clear "SEND NAK" flag	
		Start Answer Timer	
		Set "ANSWER REQ'D" flag	S1
	"SEND NAK" flag not set	Toggle "SEND ACK1/2" flag	S51
S51	Buffer full	Set "SEND WBT" flag	
		Set "WAIT FOR REP/CAN" flag	
		Start Answer Timer	
		Set "ANSWER REQ'D" flag	S6
	Buffer not full	Start Answer Timer	
		Set "ANSWER REQ'D" flag	S1

RECEIVE DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S6	"WAIT FOR CAN/REP" flag not set	----	S1
	"WAIT FOR CAN/REP" flag set	Receive a character	
	Odd parity	Ignore character	S6
	Character = SYN	Ignore character	S6
	Character = REP/CAN/ACK1/ACK2/ NAK/WBT/RM	Store character	
		Receive a character	S61
	Any other control character	Ignore character	S6
S61	Character ≠ Previous Character	Set "SEND NAK" flag	S6
	Character = REP	Perform REP routine	S6
	Character = CAN	Perform CAN routine	S6
	Character = ACK1/ACK2/NAK/WBT/RM	Perform RECEIVE CONTROL routine	S6

RECEIVE DESCRIPTION (CONTINUED)

ROUTINE	CONDITION	TASKS	NEXT STATE
REPLY	----	Set "REP REC'D" flag	
		Set "ANSWER REQ'D" flag	
		Start Answer Timer	
	Buffer full	Set "SEND WBT" flag	
		Set "WAIT FOR REP/CAN" flag	S6
	Buffer not full	Clear "WAIT FOR REP/CAN" flag	Calling State
CANCEL	----	Clear "WAIT FOR REP/CAN" flag	
		Clear "SEND ACK1" flag	
		Set "ANSWER REQ'D" flag	
		Dump buffer	
		Start Answer Timer	Calling State
RECEIVE CONTROL	"ANSWER EXP'D" flag not set	----	Calling State
	"ANSWER EXP'D" flag set	Clear "ANSWER EXP'D" flag	
	Character = WBT	Set "WBT REC'D" flag	Calling State
	Character = NAK	Set "NAK REC'D" flag	Calling State
	Character = RM	Set "RM REC'D" flag	Calling State
	Character = ACK1	Set "ACK1 REC'D" flag	Calling State
	Character = ACK2	Set "ACK2 REC'D" flag	Calling State



MODE I, Continuous RECEIVE

FIGURE 6.

TABLE II.

TRANSITION DESCRIPTION FOR TRANSMIT

STATE	CONDITION	TASKS	NEXT STATE
S1	"ANSWER REQ'D" flag set	Perform ANSWER routine	S1
	"RM REC'D" flag set	Clear "RM REC'D" flag	
		Inhibit Answer Timer	
		Send CAN sequence	
		Set "CAN SENT" flag	
		Start Answer Timer	
		Set "ANSWER EXP'D" flag	
		Clear "ACK1 EXP'D" flag	S6
	Buffer holding a block	----	S11
	No block to send	Send SYN character	S1
S11	"ETX SENT LAST" flag set	Send SOH character	S2
	"ETX SENT LAST" flag not set	Send STX character	S2
S2	----	Initialize TXTCOUNT	
	"ETX SENT LAST" flag set	----	S21
	"ETX SENT LAST" flag not set	Add DEL to BP	
		Send DEL character	S3

TRANSMIT DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S21	"RECORD TRAFFIC" flag set	Send "H" as SELect character	S3
	"OPSCOM/BULK" flag set	Send "D" as SELect character	S3
S3	"ANSWER REQ'D" flag set	Perform ANSWER routine	S3
	TXTCOUNT = 80	----	S6
	TXTCCUNT < 80	----	S31
S31	Buffer empty	Add EM character to BP	
		Send EM	S6
	Buffer not empty	Increment TXTCOUNT	
		Add character to BP	
		Send odd parity data character	S3
S4	"ANSWER REQ'D" flag set	Perform ANSWER routine	S4
	Not Last Block	Add ETB character to BP	
		Send ETB	
		Clear "ETX SENT LAST" flag	S5
	Last Block	Add ETX character to BP	
		Send ETX	
		Set "ETX SENT LAST" flag	S5

TRANSMIT DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S5	----	Send computed BP	
		Clear computed BP	
		Set "ANSWER EXP'D" flag	
		Start Answer Timer	S1
S6	"ACK1 REC'D" flag set	Clear "ACK1 REC'D" flag	S61
	"ACK2 REC'D" flag	Clear "ACK2 REC'D" flag	S62
	"RM REC'D" flag set	Clear "RM REC'D" flag	
		Inhibit Answer Timer	
		Send CAN sequence	
		Set "CAN SENT" flag	
		Start Answer Timer	
		Set "ANSWER EXP'D" flag	
		Clear "ACK1 EXP'D" flag	S6
	"WBT REC'D" flag set	Clear "WBT REC'D" flag	S6
	"NAK REC'D" flag set	Clear "NAK REC'D" flag	
		Inhibit Answer Timer	
		Prepare buffer for retransmission	S1
	Answer Timer not expired	Send SYN character	S6

TRANSMIT DESCRIPTION (CONTINUED)

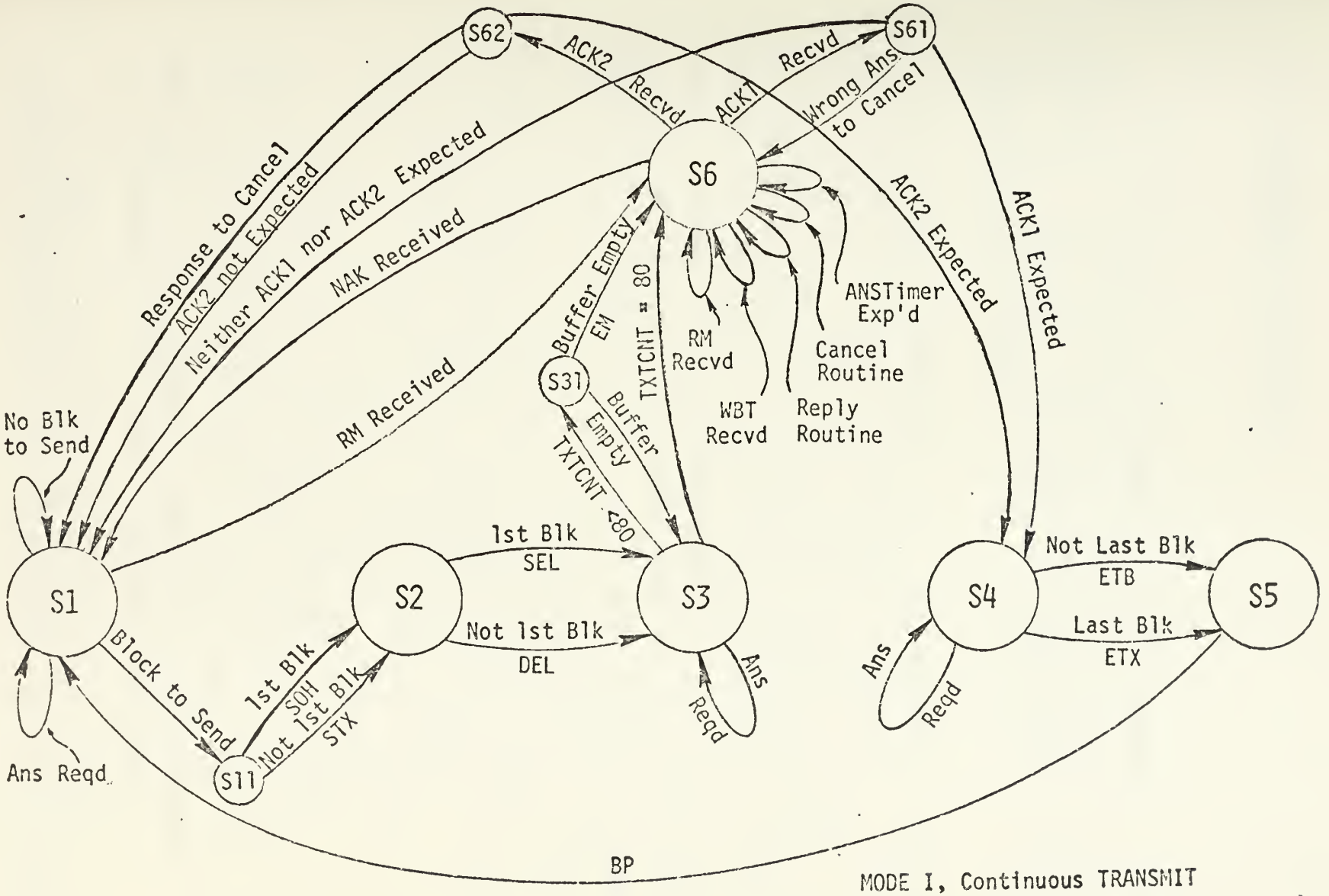
STATE	CONDITION	TASKS	NEXT STATE
	"CAN SENT" flag set	Set "NO REPLY" alarm	
	and CANCOUNTER = 3	Set CANCOUNTER = 0	
		Send CAN sequence	
		Restart Answer Timer	
		Increment CANCOUNTER	
		Set "ANSWER EXP'D" flag	S6
	"CAN SENT" flag set	Send CAN sequence	
	and CANCOUNTER < 3	Increment CANCOUNTER	
		Set "ANSWER EXP'D" flag	S6
	REPCOUNTER = 3	Set "NO REPLY" alarm	
		Set REPCOUNTER =0	
		Send REP sequence	
		Restart Answer Timer	
		Increment REPCOUNTER	
		Set"ANSWER EXP'D" flag	S6
	REPCOUNTER < 3	Send REP sequence	
		Increment REPCOUNTER	
		Set "ANSWER EXP'D" flag	S6

TRANSMIT DESCRIPTION (CONTINUED)

STATE	CONDITION	TASKS	NEXT STATE
S61	"ACK1 EXP'D" flag set	Inhibit Answer Timer	
		Clear "ACK1 EXP'D" flag	S4
	"CAN SENT" flag set	Wrong answer to CAN, so ignore	S6
	"CAN SENT" flag not set	Inhibit Answer Timer	
		Prepare buffer for retransmission	S1
S62	"ACK1 EXP'D" flag set	Inhibit Answer Timer	
		Prepare buffer for retransmission	S1
	"CAN SENT" flag set	Dump buffer	
		Clear "CAN SENT" flag	
		Set "ACK1 EXP'D" flag	
		Set CANCOUNTER = 0	
		Inhibit Answer Timer	S1
	"CAN SENT" flag not set	Inhibit Answer Timer	
		Set "ACK1 EXP'D" flag	S4

TRANSMIT DESCRIPTION (CONTINUED)

ROUTINE	CONDITION	TASKS	NEXT STATE
ANSWER	----	Clear "ANSWER REQ'D" flag	
	"SEND WBT" flag set	Clear "SEND WBT"flag	
		Send WBT sequence	Calling State
	"NAKFLG" set	Clear "NAKFLG"	
		Send NAK sequence	Calling State
	"REP REC'D" flag set	Clear "REP REC'D" flag	AR1
	"SEND ACK1" flag set	Clear "SEND ACK1" flag	
		Set "LAST RECD BLOCK ACK'D" flag	
		Send ACK1 sequence	Calling State
	"SEND ACK1" flag not set	Set "SEND ACK1" flag	
		Set "LAST RECD BLOCK ACK'D" flag	
		Send ACK2 sequence	Calling State
AR1	Last accepted block acknowledged with ACK1	Send ACK1 sequence	Calling State
	Last accepted block acknowledged with ACK2	Send ACK2 sequence	Calling State



MODE I, Continuous TRANSMIT

FIGURE 7.

E. INSTRUCTION SET DESIGN

When designing an instruction set for a special purpose such as communications control logic,two questions must be answered. They are: "What tasks need to be done?" and "What are the necessary and sufficient instructions which will accomplish those tasks?"

The tasks described in the previous transition descriptions fall into the following categories: setting and clearing flags and answer timer; clearing and incrementing counters; receiving, storing, sending and accepting characters; calculating block parity; dumping a buffer; and pushing up a buffer for retransmission. All of these actions can be accomplished by one basic transfer instruction which moves data from any one of several specified sources to one of several allowable destinations. For instance, flags can be set (or cleared) by moving a 1 (or 0) to the specified flag. Counters, buffers, and the answer timer can be handled through the use of flag type indicators. Receiving, sending, and accepting of characters is a matter of moving the character from a buffer to a register or vice versa. Storing a character can be done by moving it from the "Present Character" register to a holding (or "Previous Character") register. Block parity is calculated by the binary addition without carry of each of the bits in a character to be sent or received, with the current accumulated block parity character which is maintained in a register. This also can be accomplished with a variation of the basic transfer instruction.

In order to determine when the various transfers are to be made, a conditional jump instruction is needed to check a

particular character or flag, then either jump around an undesired area or jump to an address where the desired tasks are to be performed. An unconditional jump instruction is necessary to advance to the next state after a particular set of tasks has been accomplished. (This could use the conditional jump instruction with a condition that is always true.)

Since the transition descriptions define several subroutines, a jump-to-subroutine instruction and a return instruction are both needed. (The return instruction could be a pseudonym for the transfer instruction which moves the return address register to the current address register.)

Thus, the entire Receive and Transmit functions can be programmed using essentially three instructions (the return instruction being implemented as a transfer and the unconditional jump being implemented as an always true conditional jump.) The opcodes, formats, and detailed descriptions of these instructions are as follows:

XFR source,destination This instruction moves specified data to an allowable destination. Possible sources and their allowable destinations are:

Present Character Register (PRES)	---	HOST
		Previous
		BLKPAR
AUTODIN/NIDN Line Buffer (LINE)	---	PRES
Host Computer Buffer (HOST)	---	LINE
Any Control Character	---	LINE
		BLKPAR
Block Parity Register (BLKPAR)	---	LINE
One or Zero	---	Any flag
Current Address Register (CURRADR)	---	RTNADR
Return Address Register (RTNADR)	---	CURRADR

When a character is moved to the Block Parity register, an "exclusive OR" operation is performed. All other combinations are straight transfers of data.

JIF char/flag,condition,address This is the conditional jump instruction which jumps to a given address if a particular character is equal to (EQ) or not equal to (NOT) the contents of the Present Character register or if a certain flag is set (1) or not set (0). The address field may contain a location label or the current address register plus a specified number of instructions (e.g. CURAD + 4).

JMP address This unconditional jump uses the JIF instruction, checking for a flag which is always set.

JSR subroutine name This "jump to subroutine" instruction first stores the current address plus 1 in the Return Address register, then jumps to the specified subroutine.

RET The return instruction has no operands. It invokes the XFR instruction to move the contents of the Return Address register to the Current Address.

The above instructions have been used to program the Receive and Transmit functions as they were described in the transition descriptions of Tables I and II. The Program for Receive is presented in Table III and the Program for Transmit is presented in Table IV. The reader will notice that many jump instructions appear, since the very nature of communications control logic is the transitioning from one state to another.

TABLE III.

PROGRAM FOR RECEIVE

```
LABEL  OPCODE   OPERANDS            COMMENTS
S1     XFR      LINE,PRES           RECEIVE A CHARACTER
       JIF      PARITY,0,S1         IF PARITY ODD, IGNORE CHAR
       JIF      SYN,NOT,CURAD+5     SKIP IF PRES CHAR ≠ SYN
       XFR      1,SYNCNT            SET FLAG TO INCREMENT SYNCOUNT
       JIF      FOURS,0,CURAD+2     SKIP IF SYN COUNT ≠ 4
       XFR      1,ANS               IF 4 SYN'S, START ANSWER TIMER
       JMP      S1                  STAY IN STATE S1
       JIF      REP,EQ,S12          JUMP TO S12 IF PRES CHAR = REP
       JIF      CAN,EQ,S12          JUMP TO S12 IF PRES CHAR = CAN
       JIF      ACK1,EQ,S12         JUMP TO S12 IF PRES = ACK1
       JIF      ACK2,EQ,S12         JUMP TO S12 IF PRES = ACK2
       JIF      NAK,EQ,S12          JUMP TO S12 IF PRES CHAR = NAK
       JIF      WBT,EQ,S12          JUMP TO S12 IF PRES CHAR = WBT
       JIF      RM,EQ,S12           JUMP TO S12 IF PRES CHAR = RM
       JIF      SOH,EQ,S13          JUMP TO S13 IF PRES CHAR = SOH
       JIF      STX,EQ,S14          JUMP TO S14 IF PRES CHAR = STX
       JMP      S1                  IF IT FALLS THRU, STAY IN S1
S12    XFR      PRES,PREV           STORE CHARACTER
       XFR      LINE,PRES           RECEIVE A CHARACTER
       JIF      PREV,EQ,CURAD+3     SKIP IF PRES CHAR = PREV
       XFR      1,SNDNAK            SET "SEND NAK" FLAG
       JMP      S1                  STAY IN STATE S1
       JIF      REP,NOT,CURAD+3     SKIP IF PRES CHAR ≠ REP
       JSR      REPLY               JUMP TO REPLY SUBROUTINE
       JMP      S1                  STAY IN STATE S1
       JIF      CAN,NOT,CURAD+3     SKIP IF PRES CHAR ≠ CAN
       JSR      CANCEL              JUMP TO CANCEL SUBROUTINE
       JMP      S1                  STAY IN STATE S1
       JSR      RECON               JUMP TO RECEIVE CONTROL SUBR.
       JMP      S1                  STAY IN STATE S1
```

RECEIVE (CONTINUED)

LABEL	OPCODE	OPERANDS	COMMENTS
S13	XFR	0,ANS	INHIBIT ANSWER TIMER
	JIF	CERL,0,S1	IGNORE IF CAN/ETX ¬ RECD LAST
	JMP	S2	ACCEPT CHAR, GO TO S2
S14	XFR	0,ANS	INHIBIT ANSWER TIMER
	JIF	CERL,1,S1	IGNORE IF CAN/ETX RECD LAST
	JMP	S2	ACCEPT CHAR, GO TO S2
S2	XFR	LINE,PRES	RECEIVE A CHARACTER
	JIF	PARITY,1CURAD+4	SKIP IF PARITY EVEN
	XFR	1,WAIT	SET WAIT FOR REP/CAN FLAG
	XFR	1,ANS	START ANSWER TIMER
	JMP	S6	GO TO STATE S6
	XFR	1,CLRTXT	CLEAR TEXTCOUNT
	XFR	PRES,BLKPAR	ADD CHAR PARITY TO BP
	JIF	CERL,0,S3	IF ¬ 1ST BLK, ACCEPT, GOTO S3
S21	JIF	H,EQ,S3	IF SEL = "H", ACCEPT, GOTO S3
	JIF	D,EQ,S3	IF SEL = "D", ACCEPT, GOTO S3
	XFR	1,SNDNAK	OTHERWISE, SET SEND NAK FLAG
S3	XFR	LINE,PRES	RECEIVE A CHARACTER
	JIF	PARITY,1;CURAD+5	SKIP 4 IF PARITY EVEN
	XFR	1,TXTCNT	INCREMENT TEXTCOUNT
	XFR	PRES,BLKPAR	ADD CHAR PARITY TO BP
	XFR	PRES,HOST	PASS TEXT TO HOST PROCESSOR
	JMP	S31	GO TO S31
	JIF	ACK1,EQ,S32	GO TO S32 IF PRES = ACK1
	JIF	ACK2,EQ,S32	GO TO S32 IF PRES = ACK2
	JIF	NAK,EQ,S32	GO TO S32 IF PRES = NAK

RECEIVE (CONTINUED)

```
LABEL  OPCODE   OPERANDS              COMMENTS
       JIF      WBT,EQ,S32           GO TO S32 IF PRES = WBT
       JIF      RM,EQ,S32            GO TO S32 IF PRES = RM
       JIF      EM,NOT,CURAD+4       SKIP 3 IF CHAR ≠ EM
       XFR      PRES,BLKPAR          ADD CHAR PARITY BP
       JMP      S4                   ACCEPT CHAR, GO TO S4
       JIF      SOH,EQ,CURAD+8       SKIP 8 IF CHAR = SOH
       JIF      STX,EQ,CURAD+7       SKIP 7 IF CHAR = STX
       JIF      DEL,EQ,CURAD+6       SKIP 6 IF CHAR = DEL
       JIF      H,EQ,CURAD+5         SKIP 5 IF CHAR = H
       JIF      D,EQ,CURAD+4         SKIP 4 IF CHAR = D
       JIF      ETX,EQ,CURAD+3       SKIP 3 IF CHAR = ETX
       JIF      ETB,EQ,CURAD+2       SKIP 2 IF CHAR = ETB
       JMP      CURAD+4              SKIP IF NOT A FRAMING CHAR
       XFR      1,WAIT               SET "WAIT FOR REP/CAN" FLAG
       XFR      1,ANS                START ANSWER TIMER
       JMP      S6                   GO TO STATE S6
       XFR      1,SNDNAK             SET "SEND NAK"
       JMP      S3                   STAY IN STATE S3
S31    JIF      OVFLO,1,S4           GO TO S4 IF TEXTCOUNT = 80
       JMP      S3                   STAY IN STATE S3
S32    XFR      PRES,PREV            STORE CHARACTER
       XFR      LINE,PRES            RECEIVE A CHARACTER
       JIF      PREV,EQ,CURAD+3      SKIP 2 IF PRES =PREV
       XFR      1,SNDNAK             SET "SEND NAK" FLAG
       JMP      S3                   STAY IN STATE S3
       JSR      RECON                JUMP TO RECEIVE CONTROL SUBR.
       JMP      S3                   STAY IN STATE S3
```

RECEIVE (CONTINUED)

```
LABEL  OPCODE  OPERANDS              COMMENTS
 S4     XFR    LINE,PRES            RECEIVE A CHARACTER
        JIF    PARITY,1,CURAD+4     SKIP IF CONTROL CHAR
        XFR    1,WAIT               SET "WAIT FOR REP/CAN" FLAG
        XFR    1,ANS                START ANSWER TIMER
        JMP    S6                   GO TO STATE S6
        JIF    SYN,EQ,S4            IGNORE CHAR, STAY IN STATE S4
        JIF    REP,EQ,S41           JUMP TO S41 IF PRES = REP
        JIF    CAN,EQ,S41           JUMP TO S41 IF PRES = CAN
        JIF    ACK1,EQ,S41          JUMP TO S41 IF PRES = ACK1
        JIF    ACK2,EQ,S41          JUMP TO S41 IF PRES = ACK2
        JIF    NAK,EQ,S41           JUMP TO S41 IF PRES = NAK
        JIF    WBT,EQ,S41           JUMP TO S41 IF PRES = WBT
        JIF    RM,EQ,S41            JUMP TO S41 IF PRES = RM
        JIF    SOH,EQ,CURAD+6       SKIP 5 IF CHAR = SOH
        JIF    STX,EQ,CURAD+5       SKIP 4 IF CHAR = STX
        JIF    DEL,EQ,CURAD+4       SKIP 3 IF CHAR = DEL
        JIF    H,EQ,CURAD+3         SKIP 2 IF CHAR = H
        JIF    D,EQ,CURAD+2         SKIP 1 IF CHAR = D
        JMP    CURAD+4              JUMP IF NOT 1ST/2ND FRAME CHAR
        XFR    1,WAIT               SET "WAIT FOR REP/CAN" FLAG
        XFR    1,ANS                START ANSWER TIMER
        JMP    S6                   GO TO WAIT STATE,S6
        JIF    ETB,NOT,CURAD+4      SKIP 3 IF PRES ≠ ETB
        XFR    PRES,BLKPAR          ADD CHARACTER TO BP
        XFR    0,CERL               CLEAR "CAN/ETX REC'D LAST"
        JMP    S42                  JUMP TO S42
        JIF    ETX,NOT,CURAD+4      SKIP 3 IF PRES ≠ ETX
        XFR    PRES,BLKPAR          ADD CHARACTER TO BP
        XFR    1,CERL               SET "CAN/ETX REC'D LAST"
        JMP    S42                  JUMP TO S42
        XFR    1,SNDNAK             SET "SEND NAK"
```

RECEIVE (CONTINUED)

```
LABEL  OPCODE  OPERANDS              COMMENTS
       JMP     S4                   STAY IN STATE S4
S41    XFR     PRES,PREV            STORE CHARACTER
       XFR     LINE,PRES            RECEIVE A CHARACTER
       JIF     PREV,EQ,CURAD+3      SKIP 2 IF PRES = PREV
       XFR     1,SNDNAK             SET " SEND NAK"
       JMP     S4                   STAY IN STATE S4
       JIF     REP,NOT,CURAD+3      SKIP 2 IF PRES CHAR ≠ REP
       JSR     REPLY                JUMP TO REPLY SUBR.
       JMP     S4                   STAY IN STATE S4
       JIF     CAN,NOT,CURAD+3      SKIP 2 IF CHAR ≠ CAN
       JSR     CANCEL               JUMP TO CANCEL SUBR.
       JMP     S4                   STAY IN STATE S4
       JSR     RECON                JUMP TO RECEIVE CONTROL SUBR.
       JMP     S4                   STAY IN STATE S4
S42    JIF     LRBA,0,CURAD+3       SKIP IF LAST BLOCK NOT ACK'D
       XFR     0,LRBA               CLEAR "LAST REC'D BLOCK ACK'D"
       JMP     S5                   ACCEPT CHAR, GO TO S5
       XFR     1,WAIT               SET "WAIT FOR REP/CAN" FLAG
       XFR     1,ANS                START ANSWER TIMER
       JMP     S6                   GO TO WAIT STATE, S6
S5     XFR     LINE,PRES            RECEIVE A CHARACTER
       JIF     BLKPAR,EQ,CURAD+2    SKIP IF REC'D BP = CALC'D BP
       XFR     1,SNDNAK             SET "SEND NAK" FLAG
       XFR     BLKPAR,BLKPAR        CLEAR BLOCK PARITY
       JIF     SNDNAK,0,S51         GO TO S51 IF "SEND NAK" = 0
       XFR     1,DMPBUF             SET FLAG TO DUMP BUFFER
       XFR     1,NAKFLG             SET NAK FLAG FOR TRANSMITTER
       XFR     0,SNDNAK             CLEAR LOCAL "SEND NAK" FLAG
       XFR     1,ANS                START ANSWER TIMER
       XFR     1,ANREQ              SET ANSWER REQUESTED FLAG
       JMP     S1                   GO TO STATE S1
```

RECEIVE (CONTINUED)

```
LABEL  OPCODE   OPERANDS              COMMENTS
 S51    JIF    SNDACK1,0,CURAD+3     SKIP IF ACK2 SENT LAST
        XFR    0,SNDACK1             CLEAR "SEND ACK1" FLAG
        JMP    CURAD+2
        XFR    1,SNDACK1             SET "SEND ACK1" FLAG
        JIF    BUFULL,0,CURAD+6      SKIP 5 IF BUFFER NOT FULL
        XFR    1,SNDWBT              SET "SEND WBT" FLAG
        XFR    1,WAIT                SET "WAIT FOR REP/CAN" FLAG
        XFR    1,ANS                 START ANSWER TIMER
        XFR    1,ANREQ               SET "ANSWER REQ'D" FLAG
        JMP    S6                    GO TO WAIT STATE,S6
        XFR    1,ANS                 START ANSWER TIMER
        XFR    1,ANREQ               SET "ANSWER REQ'D" FLAG
        JMP    S1                    GO TO STATE S1
 S6     JIF    WAIT,0,S1             GO TO S1 IF WAIT FLAG CLEAR
        XFR    LINE,PRES             RECEIVE A CHARACTER
        JIF    PARITY,0,S6           IGNORE ODD CHAR, STAY IN S6
        JIF    SYN,NOT,CURAD+5       SKIP IF NOT SYN CHAR
 S61    XFR    1,SYNCNT              INCREMENT SYN COUNTER
        JIF    FOURS,0,S6            STAY IN S6 IF SYNCOUNT ≠ 4
        XFR    1,ANS                 START ANSWER TIMER
        JMP    S6                    STAY IN STATE S6
        JIF    REP,EQ,S62            JUMP TO S62 IF PRES = REP
        JIF    CAN,EQ,S62            JUMP TO S62 IF PRES = CAN
        JIF    ACK1,EQ,S62           JUMP TO S62 IF PRES = ACK1
        JIF    ACK2,EQ,S62           JUMP TO S62 IF PRES = ACK2
        JIF    NAK,EQ,S62            JUMP TO S62 IF PRES = NAK
        JIF    WBT,EQ,S62            JUMP TO S62 IF PRES = WBT
        JIF    RM,EQ,S62             JUMP TO S62 IF PRES = RM
        JMP    S6                    IF NONE OF ABOVE STAY IN S6
```

RECEIVE (CONTINUED)

```
LABEL   OPCODE   OPERANDS              COMMENTS
 S62     XFR     PRES,PREV             STORE PRES CHAR IN PREV
         XFR     LINE,PRES             RECEIVE A CHARACTER
         JIF     PREV,EQ,CURAD+3       SKIP 2 IF PRES = PREV
         XFR     1,SNDNAK              IF NOT EQUAL SET "SEND NAK"
         JMP     S6                    STAY IN STATE S6
         JIF     REP,NOT,CURAD+3       SKIP 2 IF CHAR ≠ REP
         JSR     REPLY                 JUMP TO REPLY SUBR.
         JMP     S6                    STAY IN STATE S6
         JIF     CAN,NOT,CURAD+3       SKIP 2 IF CHAR ≠ CAN
         JSR     CANCEL                JUMP TO CANCEL SUBR.
         JMP     S6                    STAY IN STATE S6
         JSR     RECON                 JUMP TO RECEIVE CONTROL SUBR.
         JMP     S6                    STAY IN STATE S6
REPLY    XFR     1,REPRECD             SET "REP REC'D" FLAG
         XFR     1,ANREQ               SET "ANSWER REQ'D" FLAG
         XFR     1,ANS                 START ANSWER TIMER
         JIF     BUFULL,0,CURAD+4      SKIP 3 IF BUFFER NOT FULL
         XFR     1,SNDWBT              SET "SEND WBT" FLAG
         XFR     1,WAIT                SET "WAIT FOR REP/CAN" FLAG
         JMP     S6                    JUMP TO WAIT STATE, S6
         XFR     0,WAIT                CLEAR "WAIT FOR REP/CAN"
         RET                           RETURN TO CALLING STATE
CANCEL   XFR     0,WAIT                CLEAR "WAIT FOR REP/CAN" FLAG
         XFR     0,SNDACK1             CLEAR "SEND ACK1" FLAG
         XFR     1,CANRECD             SET "CAN REC'D" FLAG
         XFR     1,ANREQ               SET "ANSWER REQ'D" FLAG
         XFR     1,DMPBUF              SET FLAG TO DUMP BUFFER.
         XFR     1,ANS                 START ANSWER TIMER
         RET                           RETURN TO CALLING STATE
```

RECEIVE (CONTINUED)

```
LABEL  OPCODE   OPERANDS              COMMENTS
RECON  JIF      ANEXPD,1,CURAD+2      SKIP IF "ANSWER EXP'D"
       RET                            RETURN TO CALLING STATE
       JIF      WBT,NOT,CURAD+4       SKIP 3 IF CHAR≠ WBT
       XFR      1,WBTRECD             SET "WBT REC'D" FLAG
       XFR      1,LBANSD              SET "LAST BLOCK ANSW'D" FLAG
       RET                            RETURN TO CALLING STATE
       JIF      NAK,NOT,CURAD+3       SKIP 2 IF CHAR ≠ NAK
       XFR      1,NAKRECD             SET "NAK REC'D" FLAG
       RET                            RETURN TO CALLING STATE
       JIF      RM,NOT,CURAD+4        SKIP 3 IF CHAR ≠ RM
       XFR      1,RMRECD              SET "RM REC'D" FLAG
       XFR      1,LBANSD              SET "LAST BLOCK ANSW'D" FLAG
       RET                            RETURN TO CALLING STATE
       JIF      ACK1,NOT,CURAD+3      SKIP 2 IF CHAR ≠ ACK1
       XFR      1,ACK1RCD             SET "ACK1 REC'D" FLAG
       RET                            RETURN TO CALLING STATE
       XFR      1,ACK2RCD             SET "ACK2 REC'D" FLAG
       RET                            RETURN TO CALLING STATE
```

TABLE IV.

PROGRAM FOR TRANSMIT

```
LABEL  OPCODE  OPERANDS              COMMENTS
 S1     JIF    ANREQ,0,CURAD+2      SKIP IF "ANSWER REQ'D" NOT SET
        JSR    ANSWER               JUMP TO ANSWER ROUTINE
        JIF    RMRECD,1,S11         GO TO S11 IF "RM REC'D" SET
        JIF    BUFMT,0,S12          GO TO S12 IF BLOCK TO SEND
        XFR    SYN,LINE             SEND SYN CHARACTER
        JMP    S1                   STAY IN STATE S1
 S11    JIF    ANRECD,1,CURAD+3     JUMP IF "ANSWER REC'D"
        XFR    SYN,LINE             SEND SYN CHARACTER
        JMP    S1                   STAY IN STATE S1
        XFR    0,RMRECD             CLEAR "RM REC'D" FLAG
        XFR    CAN,LINE             SEND THE CANCEL
        XFR    CAN,LINE             CHARACTER SEQUENCE
        XFR    1,CANSNT             SET "CAN SENT" FLAG
        XFR    1,ANS                START ANSWER TIMER
        XFR    1,ANEXPD             SET "ANSWER EXP'D" FLAG
        XFR    0,ACK1EXP            CLEAR "ACK1 EXP'D" FLAG
        JMP    S6                   GO TO WAIT STATE, S6
 S12    JIF    ETXRL,0,CURAD+3      JUMP IF NOT FIRST BLOCK
        XFR    SOH,LINE             SEND SOH CHARACTER
        JMP    S2                   GO TO STATE S2
        XFR    STX,LINE             SEND STX CHARACTER
 S2     XFR    1,CLRTXT             SET FLAG TO CLEAR TEXTCOUNT
        JIF    ETXRL,1,S21          GO TO S21 IF ETX REC'D LAST
        XFR    DEL,BLKPAR           ADD DEL CHAR TO BLOCK PARITY
        XFR    DEL,LINE             SEND DEL CHARACTER
        JMP    S3                   GO TO STATE S3
 S21    JIF    RTRAFF,0,CURAD+3     JUMP IF NOT RECORD TRAFFIC
        XFR    H,LINE               SEND "H" AS SELECT CHAR
        JMP    S3                   JUMP TO STATE S3
        XFR    D,LINE               SEND "D" AS SELECT CHAR
```

TRANSMIT (CONTINUED)

```
LABEL  OPCODE   OPERANDS            COMMENTS
 S3      JIF   ANREQ,0,CURAD+2     SKIP IF "ANSWER REQ'D" NOT SET
         JSR   ANSWER              PERFORM ANSWER SUBROUTINE
         JIF   OVFLO,1,S6          GO TO S6 IF TEXTCOUNT = 80
 S31     JIF   BUFMT,1,CURAD+4     SKIP 3 IF BUFFER EMPTY
         XFR   EM,BLKPAR           ADD EM CHARACTER TO BP
         XFR   EM,LINE             SEND EM CHARACTER
         JMP   S6                  GO TO CONTROL STATE, S6
         XFR   1,TXTCNT            INCREMENT TEXTCOUNT
         XFR   HOST,LINE           SEND ODD PARITY DATA CHARACTER
         XFR   HOST,BLKPAR         ADD CHAR TO BLOCK PARITY
         JMP   S3                  STAY IN STATE S3
 S4      JIF   ANREQ,0,CURAD+2     SKIP IF "ANSWER REQ'D" NOT SET
         JSR   ANSWER              PERFORM ANSWER ROUTINE
         JIF   EOMFLG,1,CURAD+5    SKIP IF LAST BLOCK
         XFR   ETB,BLKPAR          ADD ETB CHARACTER TO BP
         XFR   ETB,LINE            SEND ETB CHARACTER
         XFR   0,ETXSL             CLEAR "ETX SENT LAST" FLAG
         JMP   S5                  GO TO STATE S5
         XFR   ETX,BLKPAR          ADD ETX CHARACTER TO BP
         XFR   ETX,LINE            SEND ETX CHARACTER
         XFR   1,ETXSL             SET "ETX SENT LAST" FLAG
 S5      XFR   BLKPAR,LINE         SEND COMPUTED BLOCK PARITY
         XFR   BLKPAR,BLKPAR       CLEAR BLOCK PARITY
         XFR   1,ANEXPD            SET "ANSWER EXPD'D" FLAG
         XFR   1,ANS               START ANSWER TIMER
         JMP   S1                  GO TO STATE S1
```

TRANSMIT (CONTINUED)

```
LABEL  OPCODE   OPERANDS              COMMENTS
 S6      JIF   ACK1RCD,1,S61         GO TO S61 IF "ACK1 REC'D"
         JIF   ACK2RCD,1,S62         GO TO S62 IF "ACK2 REC'D"
         JIF   RMRECD,0,CURAD+9      SKIP IF "RM REC'D" NOT SET
         XFR   0,ANS                 INHIBIT ANSWER TIMER
         XFR   0,RMRECD              CLEAR "RM REC'D" FLAG
         XFR   CAN,LINE              SEND CAN
         XFR   CAN,LINE              CHARACTER SEQUENCE
         XFR   1,ANS                 START ANSWER TIMER
         XFR   1,ANEXPD              SET "ANSWER EXP'D" FLAG
         XFR   0,ACK1EXP             CLEAR "ACK1 EXP'D" FLAG
         JMP   S6                    STAY IN STATE S6
         JIF   WBTRCD,0,CURAD+3      SKIP IF "WBT REC'D" NOT SET
         XFR   0,WBTRCD              CLEAR "WBT REC'D" FLAG
         JMP   S6                    STAY IN STATE S6
         JIF   NAKRCD,0,CURAD+5      SKIP IF "NAK REC'D" NOT SET
         XFR   0,ANS                 INHIBIT ANSWER TIMER
         XFR   0,NAKRCD              CLEAR "NAK REC'D" FLAG
         XFR   1,PUSHUP              SET FLAG TO PUSH UP BUFFER
         JMP   S1                    GO TO STATE S1
         JIF   TYM,0,CURAD+3         SKIP IF ANS TIMER NOT EXPIRED
         XFR   SYN,LINE              SEND SYN CHARACTER
         JMP   S6                    STAY IN STATE S6
         JIF   CANSNT,0,CURAD+10     SKIP IF "CAN SENT" NOT SET
         JIF   CAN3,0,CURAD+3        SKIP IF <3 CANCEL SEQS SENT
         XFR   1,NOREP               SET "NO REPLY FLAG
         XFR   1,CLRCAN              SET FLAG TO CLEAR CAN COUNT
         XFR   CAN,LINE              SEND CANCEL
         XFR   CAN,LINE              CHARACTER SEQUENCE
         XFR   1,ANS                 RESTART ANSWER TIMER
         XFR   1,CANCNT              SET FLAG TO INCREMENT CANCOUNT
         XFR   1,ANEXPD              SET"ANSWER EXP'D" FLAG
         JMP   S6                    STAY IN STATE S6
```

TRANSMIT (CONTINUED)

```
        JIF   RM3,0,CURAD+3      SKIP IF <3 RM SEQS SENT
        XFR   1,NOREP            SET"NO REPLY" ALARM FLAG
        XFR   1,CLRREP           SET FLAG TO CLEAR REP COUNT
        XFR   REP,LINE           SEND REPLY
        XFR   REP,LINE           CHARACTER SEQUENCE
        XFR   1,ANS              RESTART ANSWER TIMER
        XFR   1,REPCNT           SET FLAG TO INCREMENT REPCOUNT
        XFR   1,ANEXPD           SET "ANSWER EXP'D" FLAG
        JMP   S6                 STAY IN STATE S6
S61     JIF   ACK1EXP,0,CURAD+4  SKIP IF "ACK1 EXP'D" NOT SET
        XFR   0,ANS              INHIBIT ANSWER TIMER
        XFR   0,ACK1EXP          CLEAR "ACK1 EXP'D" FLAG
        JMP   S4                 GO TO STATE S4
        JIF   CANSNT,S6          IF "CAN SENT", GO TO S6
        XFR   0,ANS              INHIBIT ANSWER TIMER
        XFR   1,PUSHUP           SET FLAG TO PUSH UP BUFFER
        XFR   S1                 GO TO STATE S1
S62     JIF   ACK1EXP,0,CURAD+4  SKIP IF "ACK1 EXP'D" NOT SET
        XFR   0,ANS              INHIBIT ANSWER TIMER
        XFR   1,PUSHUP           SET FLAG TO PUSH UP BUFFER
        JMP   S1                 GO TO STATE S1
        JIF   CANSNT,0,CURAD+5   SKIP IF "CAN SENT" NOT SET
        XFR   1,DMPBUF           SET FLAG TO DUMP BUFFER
        XFR   1,CLRCAN           SET FLAG TO CLEAR CAN COUNTER
        XFR   0,ANS              INHIBIT ANSWER TIMER
        JMP   S1                 GO TO STATE S1
        XFR   0,ANS              INHIBIT ANSWER TIMER
        XFR   1,ACK1EXP          SET "ACK1 EXP'D" FLAG
        JMP   S4                 GO TO STATE S4
```

TRANSMIT (CONTINUED)

```
LABEL  OPCODE   OPERANDS              COMMENTS
ANSWER XFR      0,ANREQ               CLEAR "ANSWER REQ'D" FLAG
       JIF      SNDWBT,0,CURAD+5      SKIP IF "SEND WBT" NOT SET
       XFR      0,SNDWBT              CLEAR "SEND WBT" FLAG
       XFR      WBT,LINE              SEND WBT
       XFR      WBT,LINE              CHARACTER SEQUENCE
       RET                            RETURN TO CALLING STATE
       JIF      NAKFLG,0,CURAD+5      SKIP IF NAK FLAG NOT SET
       XFR      0,NAKFLG              CLEAR "SEND NAK" FLAG
       XFR      NAK,LINE              SEND NAK
       XFR      NAK,LINE              CHARACTER SEQUENCE
       RET                            RETURN TO CALLING STATE
       JIF      REPRECD,1,AR1         IF "RM REC'D", GO TO AR1
       JIF      SNDACK1,0,CURAD+6     SKIP IF "SEND ACK1" NOT SET
       XFR      0,SNDACK1             CLEAR "SEND ACK1" FLAG
       XFR      1,LRBA                SET "LAST REC'D BLOCK ACK'D"
       XFR      ACK1,LINE             SEND ACK1
       XFR      ACK1,LINE             CHARACTER SEQUENCE
       RET                            RETURN TO CALLING STATE
       XFR      1,SNDACK1             SET "SEND ACK1" FLAG
       XFR      1,LRBA                SET "LAST REC'D BLOCK ACK'D"
       XFR      ACK2,LINE             SEND ACK2
       XFR      ACK2,LINE             CHARACTER SEQUENCE
       RET                            RETURN TO CALLING STATE
 AR1   JIF      SNDACK1,1,CURAD+4     SKIP IF LAST BLK WAS ACK2
       XFR      ACK1,LINE             SEND ACK1
       XFR      ACK1,LINE             CHARACTER SEQUENCE
       RET                            RETURN TO CALLING STATE
       XFR      ACK2,LINE             SEND ACK2
       XFR      ACK2,LINE             CHARACTER SEQUENCE
       RET                            RETURN TO CALLING STATE
```

F. IMPLEMENTATION PLAN

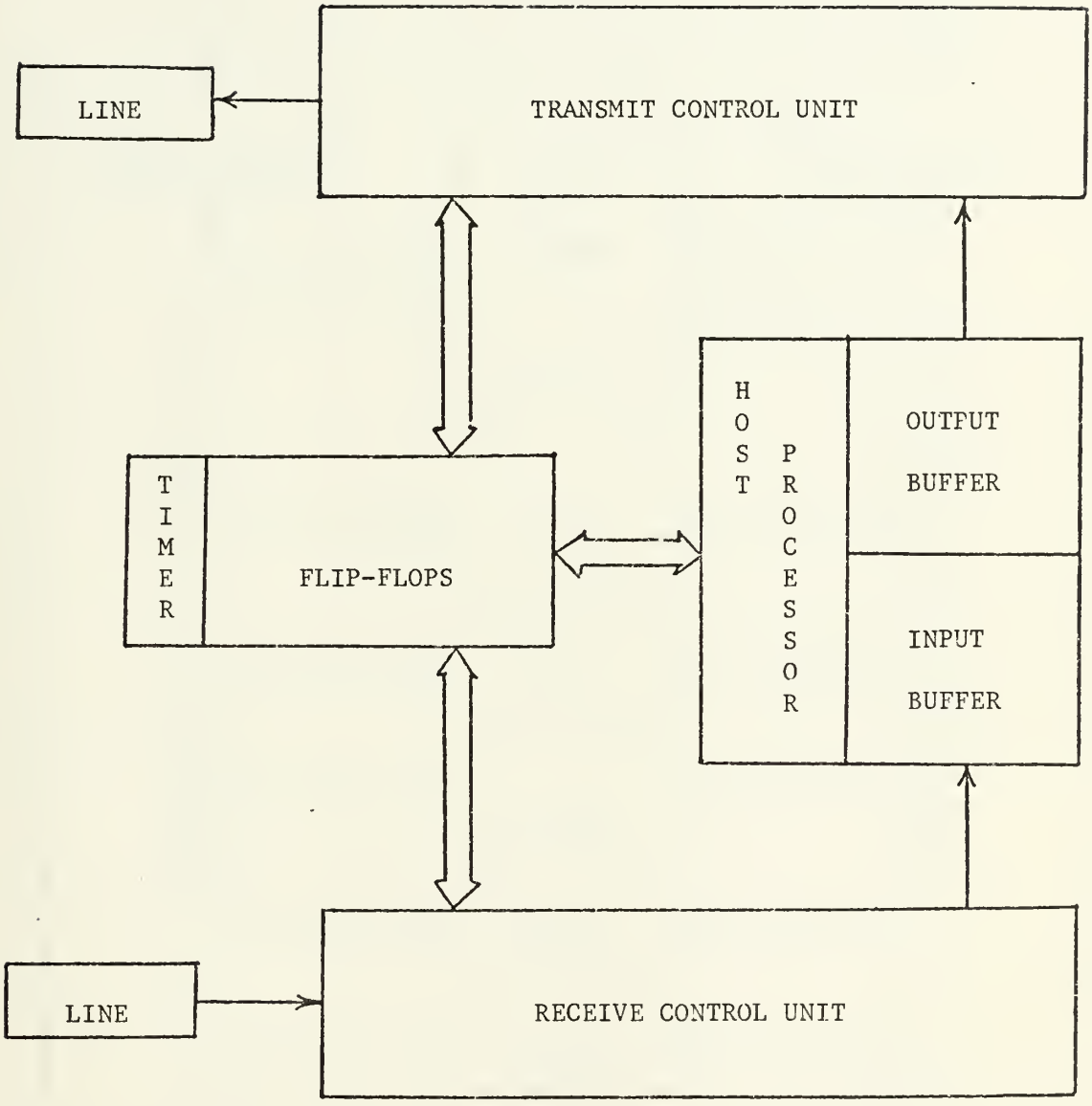
To set the scene for a possible implementation scheme, some interface assumptions must be specified. First of all, the Receive and Transmit functions interact with the Host buffer and Line registers, character by character. Control characters are removed by the Receive function and added by the Transmit function; only the text characters are passed between the Receive and Transmit functions and the Host (no more than 80 per block). When a text character is taken from the Host buffer register by the Transmit function, the next character in the buffer is automatically placed in the register. Since the Host processor must obviously know when it is working on the last block of a message, it can set an "end of message" flag for the Transmit control. On the other hand, the Host must check flags set by the Receive or Transmit logic in order to determine when to dump (ignore) a buffer or prepare a buffer for retransmission. The Transmit Line buffer register must delay execution of a current "send" until a previous "send" has been completed. (This requirement allows the program to issue two successive XFR instructions with "LINE" as the destination operand, as is needed for the two-character control sequences.)

The Transmit and Receive functions require a total of 35 flags or indicators. Fourteen of these are used in common by both. Eight flags are local to the Receive logic and thirteen are local to Transmit. Eight of these same 35 flags are referenced by the Host processor.

There are seven registers. Six are used as both source and destination. One, the Previous Character register, is a destination register only.

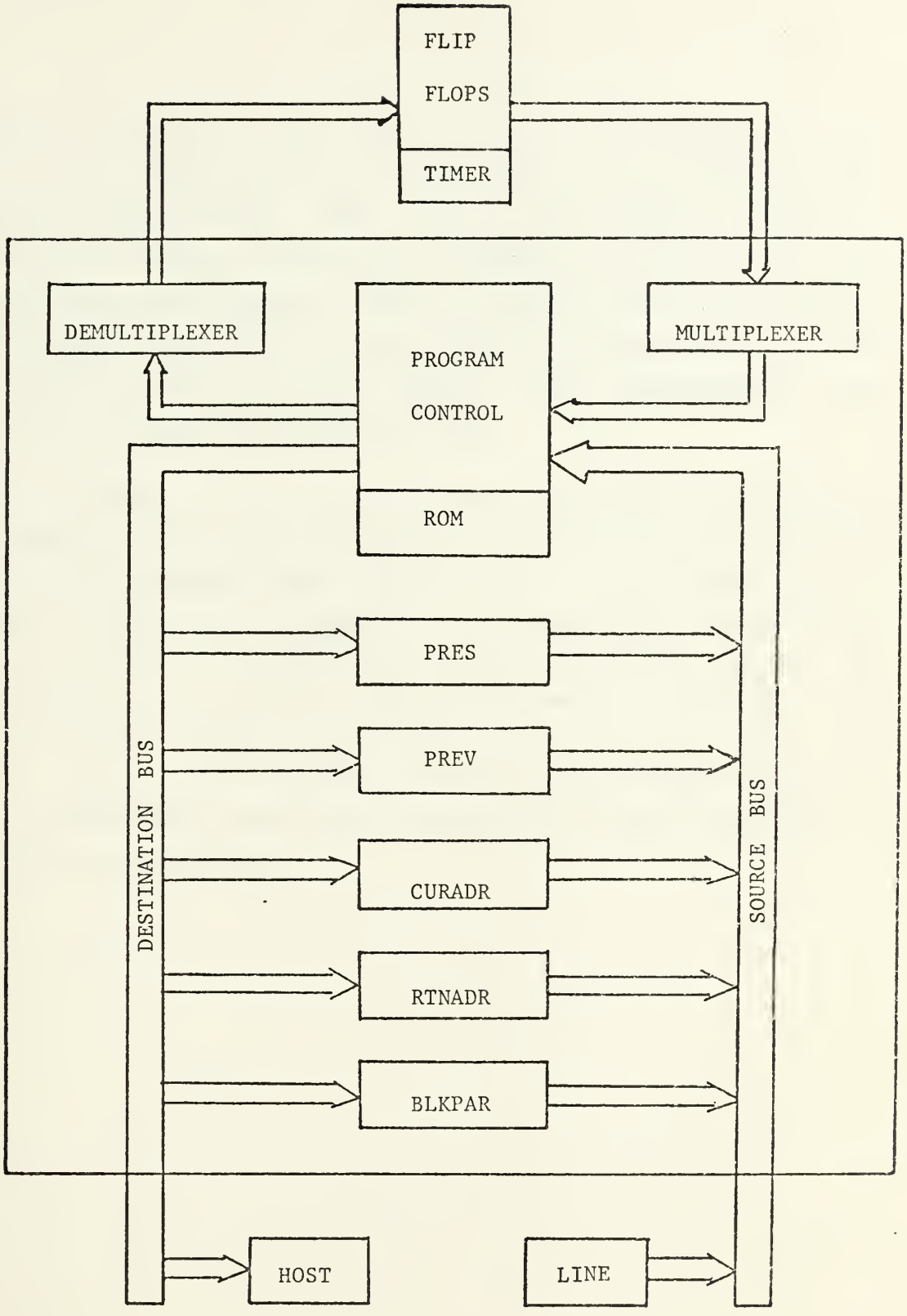
The programs as written in Tables III and IV use 220 addresses for the Receive logic and 146 addresses for the Transmit logic.

If the control logic functions are to be implemented in random logic hardware, a bit configuration for the designed instruction set must first be derived. Then the programs can be put into a Read Only Memory (ROM) and be called out by a program control unit as needed. One possible implementation scheme is presented in Figures 8 and 9. Figure 8 is a simple diagram showing the interaction of the Transmit and Receive control functions with the Host processor, the flag flip-flops and timer, and the AUTODIN/NIDN Line buffers. Figure 9 represents the Receive control logic unit. The registers are all connected by buses, which transfer the data from one register to another as directed by the program control logic. The flip-flops are set and cleared via a demultiplexer and checked by means of a multiplexer. The block parity register (BLKPAR) has a built-in exclusive "or" function, i.e. whenever any data is transferred to BLKPAR, it is automatically exclusive "or"ed with the current contents of BLKPAR. Likewise, the return address register (RTNADR) automatically adds one to the current address register (CURADR) data as it is entered. (CURADR is the only source for data being transferred to RTNADR.)



CONTROL UNIT INTERFACE

FIGURE 8.



RECEIVE CONTROL UNIT

FIGURE 9.

This arrangement will accomplish all the actions called for by the instruction set. No figure is provided for the Transmit control logic unit since the diagram would be identical to Figure 8 with the exception that the data lines between the Host register and the bus, and the Line register and the bus, would point in the opposite direction, i.e. the data would transfer from the HOST to the Destination bus, and from the Source bus to the LINE.

An alternative to the above implementation plan would be the use of a conventional general-purpose microprocessor to emulate the designed instruction set. This scheme would provide much more flexibility than the first and would be good for testing the validity of the control logic for completeness and timing. If changes become necessary, they could be easily made by modifying the program. With the "state of the art" rapidly advancing, microprocessors may soon be developed which would provide the speed required of a backup control unit.

V. SUMMARY AND CONCLUSIONS

A step-by-step reduction of AUTODIN/NIDN communication protocol to programmed control logic has been presented. Following a general background description of the AUTODIN line discipline, a detailed analysis of the Transmit and Receive functions of AUTODIN (as modified for NIDN and OSIS) provided the basis for state and transition descriptions. These descriptions, in turn, led to transition tables and state diagrams. A special instruction set, capable of accomplishing any of the described tasks, was then developed and used to program the Transmit and Receive control functions from the transition tables and state diagrams. Two possible implementation schemes were outlined.

Although this document deals specifically with the AUTODIN/NIDN line disciplines used by OSIS, the method of approach applied to reducing this protocol to programmed control logic is general enough to be employed in reducing other communication systems. The approach is simple and straightforward and the resulting sequence of tables, diagrams, and programs are easy to follow and understand. Especially to be emphasized is the ease of making changes with this method. If it is necessary to alter the Receive or Transmit control function in any way, it need only be determined which state is effected, then locate that state in the transition table, state diagram, and program and make the appropriate additions or changes to all three.

BIBLIOGRAPHY

1. Chu, Yaohan , Digital Computer Design Fundamentals , McGraw-Hill, 1962.

2. Defense Communications Agency Circular 370-D175-1, DCS Autodin Interface and Control Criteria, October 1970.

3. Dietmeyer, Donald L., Logic Design of Digital Systems, Allyn and Bacon, Inc., 1971.

4. Joint Chiefs of Staff Publication JANAP 128 (e), Automatic Digital Network (AUTODIN) Operating Procedures, June 1973.

5. Martin, James, Telecommunications Network Organization, Prentice-Hall, 1970.

6. McCluskey, E. J., Jr. and Bartee, T. C., A Survey of Switching Circuit Theory, McGraw-Hill, 1962.

7. McManis, Robert Bruce, Computerized Management Tools for Use in the Analysis of Autodin Automatic Switching Center and Associated Tributaries, Naval Postgraduate School Master's Thesis, March 1973.

8. Mills, David L., Topics in Computer Communications Systems, Concomp Project Technical Report 20,University of Michigan, May 1969.

9. Naval Electronics Laboratory Center Technical Note 2211, Implementation Plan for Autodin/Datanet 355 Interface, a tentative and unpublished working paper by Keele, R.V. and Stephenson, G. R., 2 November 1972.

10. Naval Electronics Laboratory Center Technical Note 2579 Revision B, OSIS Communications Line Discipline and Message Exchange Conventions, a tentative and unpublished working paper by WWMMCS Office, May 1975.

11. Naval Electronics Laboratory Center Technical Note 2822, OSIS Communications Program Description for ONH Module (OIC), a tentative and unpublished working paper by Stanley, G.V., 4 November 1974.

12. NAVSHIPS 0967-324-0100, Digital Subscriber Terminals AN/FYA-71 (V) 1 Through AN/FYA-71 (V) 6 Maintenance Manual, March 1969.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Chairman, Code 72 Computer Science Group Naval Postgraduate School Monterey, California 93940	1
4.	Professor V. Michael Powers, Code 72Pw Naval Postgraduate School Monterey, California 93940	1
5.	Professor Rudolf Panholzer, Code 52Pz Naval Postgraduate School Monterey, California 93940	1
6.	LCDR Jane F. Renninger, USN Naval Electronics Laboratory Center San Diego, California 92152	1
7.	Naval Electronics Laboratory Center Code 1130 San Diego, California 92152	1

8. Naval Electronics Laboratory Center 1
Code 5200
M. A. Lamendola
San Diego, California 92152

161149
Thesis
R345 Renninger
c.1 A reduction of
AUTODIN/NIDN line disci-
plines to programmed
control logic.